INGENIERÍA DE APLICACIONES

Calidad de Software

Dra. María Luján Ganuza
mlg@cs.uns.edu.ar
DCIC - Depto. de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Bahía Blanca
2018

Temario

Calidad de Software.

Atributos de Calidad de Software.

Estándares de Calidad de Software.

Revisiones e Inspecciones.

Medidas y Métricas de Software.

Calidad de Software

Con la creciente automatización en la vida cotidiana, cada vez más personas entran en contacto con sistemas de software, y la calidad de esos sistemas es una gran preocupación.

La calidad no se puede añadir en el último momento. Tiene que ser construida desde el principio.

Calidad de Software

- Uno de esos factores clave que comparten las compañías exitosas es el compromiso con la calidad.
- Debido a la gran complejidad de los productos de software y los frecuentes cambios que deben incorporarse durante el desarrollo del software, se necesita atención continua y evaluación de la calidad del producto.
- Esta necesidad se ve agravada por la creciente penetración de la tecnología de software en la vida cotidiana.

Calidad de Software

- Un ejemplo de lo que puede suceder si el software contiene errores, se conoce como "Malfunction 54“.

Terac-25

Calidad de Software

- El compromiso con la calidad en el desarrollo de software no solo da sus frutos, sino que es una necesidad absoluta.

- Si **imponemos** ciertos requisitos de calidad en el producto o proceso, podemos idear técnicas y procedimientos para **asegurar o probar** que el producto o proceso efectivamente se ajuste a esos requisitos.

Calidad de Software

- La **calidad del software** es una noción bastante elusiva. Diferentes personas tendrán diferentes perspectivas sobre la calidad de un sistema de software.
- Un verificador del sistema puede ver la calidad como *cumplimiento de requisitos*, mientras que un usuario puede verlo como aptitud *para el uso*.
- Ambos puntos de vista son válidos, pero no necesitan coincidir. De hecho, probablemente no lo harán.

Calidad de Software

La gestión de calidad total (TQM) aboga por una visión ecléctica:

La calidad es la búsqueda de la excelencia en todo.

Calidad de Software

En la industria de la manufactura una definición de **calidad**, que se basa en

la conformidad con una especificación detallada del producto y la noción de tolerancias.

Los productos nunca cumplirán exactamente una especificación, por lo que se permitió cierta tolerancia.

Si el producto es "casi correcto", se clasifica como aceptable.

Calidad de Software

- La calidad del software no es directamente comparable con la calidad en la industria de la manufactura.
- La idea de tolerancias no es aplicable a los sistemas digitales.
- Puede ser imposible llegar a una conclusión **objetiva** sobre si un sistema de software cumple o no con su especificación.

Calidad de Software

En el desarrollo de Software:

Es difícil escribir especificaciones de software completas y sin ambigüedades.

Los desarrolladores y los clientes pueden interpretar los requisitos de diferentes maneras y puede ser imposible llegar a un acuerdo sobre si el software se ajusta o no a su especificación.

Calidad de Software

En el desarrollo de Software:

Las especificaciones generalmente integran requisitos de varias clases de partes interesadas.

Estos requisitos son inevitablemente un compromiso y pueden no incluir los requisitos de todos los grupos de partes interesadas. Por lo tanto, los interesados excluidos pueden percibir el sistema como un sistema de mala calidad, aunque implementa los requisitos acordados.

Calidad de Software

En el desarrollo de Software:

Es imposible medir ciertas características de calidad directamente

y por lo que no se pueden especificar de una manera inequívoca.

Calidad de Software

- ¿Se han seguido los estándares de programación y documentación en el proceso de desarrollo?
- ¿El software ha sido probado adecuadamente?
- ¿Es el software lo suficientemente confiable para ser puesto en uso?
- ¿El rendimiento del software es aceptable para un uso normal?
- ¿Es el software utilizable?
- ¿El software está bien estructurado y es comprensible?

Calidad de Software

- Hay una suposición general en la gestión de la calidad del software que el sistema será probado contra sus requisitos.
- La **calidad subjetiva** de un sistema de software se basa principalmente en sus características **no funcionales.**
  - Si la funcionalidad del software no es la esperada, los usuarios a menudo lo solucionarán y encontrarán otras formas de hacer lo que quieran.
  - Si el software no es confiable o es demasiado lento, entonces es prácticamente imposible para ellos lograr sus objetivos.

Atributos de Calidad de Software

- Seguridad
- Comprensibilidad
- Portabilidad
- Testeabilidad
- Usabilidad
- Confiabilidad
- Adaptabilidad
- Reusabilidad
- Resiliencia
- Eficiencia
- Robustez
- Capacidad de Aprendizaje

Atributos de Calidad de Software

Seguridad

La capacidad del producto de software para proteger información y datos para:

- que las personas o sistemas no autorizados no puedan leerlos o modificarlos y
- no se les niega el acceso a personas o sistemas autorizados.

Atributos de Calidad de Software

Comprensibilidad

La comprensibilidad define qué tan fácil es para el usuario entender un producto de software.

Las interfaces y operaciones del sistema deben ser fáciles de entender y su aplicabilidad debe ser evidente.

Atributos de Calidad de Software

Portabilidad

La capacidad del producto de software para ser transferido de un entorno a otro.

Es necesaria la abstracción generalizada entre la lógica de la aplicación y las interfaces del sistema.

Atributos de Calidad de Software

Testeabilidad

Capacidad de seguir la ejecución del programa y su la depuración.

Depende de la modularidad del código y su estructura.

Los programas modulares y bien estructurados resultan más adecuados para las pruebas sistemáticas y por etapas que los programas monolíticos y no estructurados.

Atributos de Calidad de Software

Usabilidad

La capacidad del producto de software para ser entendido, aprendido, utilizado y ser atractivo para el usuario, cuando se utiliza bajo condiciones específicas.

Atributos de Calidad de Software

Confiabilidad

La capacidad del producto de software para mantener un nivel de rendimiento específico cuando se utiliza bajo condiciones específicas.

Un sistema de software puede considerarse confiable si las prueba producen una tasa baja de errores.

La tasa de error depende de la frecuencia de las entradas y de la probabilidad de que una entrada individual genere un error.

Atributos de Calidad de Software

Adaptabilidad

Capacidad de un sistema de adaptarse automáticamente a sus usuarios.

Capacidad para tolerar que los usuarios puedan personalizar sustancialmente el sistema mediante la personalización de actividades.

Atributos de Calidad de Software

Reusabilidad

El grado en que un programa (o partes del mismo) puede ser reutilizado en otras aplicaciones.

Atributos de Calidad de Software

Resiliencia

La capacidad de un sistema de software de recuperarse de ciertos tipos de fallas y seguir siendo funcional desde la perspectiva del cliente.

También se puede llamar Recuperabilidad.

El software es resiliente si sus funciones críticas se pueden recuperar de una falla.

Atributos de Calidad de Software

Eficiencia

La capacidad del producto de software para proporcionar un rendimiento adecuado, en relación con la cantidad de recursos utilizados, en las condiciones establecidas.

Atributos de Calidad de Software

Robustez

La capacidad de un sistema de continuar funcionando en presencia de entradas no válidas o condiciones ambientales estresantes.

La capacidad de los sistemas de software para reaccionar adecuadamente ante condiciones anormales.

Atributos de Calidad de Software

Capacidad de Aprendizaje

La capacidad de un producto de software para permitir que el usuario aprenda a usarlo.

Atributos de Calidad de Software

- Seguridad
- Comprensibilidad
- Portabilidad
- Resiliencia
- Eficiencia
- Robustez

No es posible optimizar un sistema para todos estos atributos

- Adaptabilidad
- Reusabilidad

Atributos de Calidad de Software

- El plan de calidad debe definir los atributos de calidad más importantes para el software que se está desarrollando.
- Puede ser que la eficiencia sea crítica y otros factores deben sacrificarse para lograr esto.
- El plan también debe incluir una definición del proceso de evaluación de la calidad.

Estándares de Calidad de Software

- Los estándares definen un conjunto de criterios que guían la forma en que se aplican procedimientos y metodologías al software desarrollado.
- La certificación de calidad permite una valoración independiente de la organización, donde se demuestra la capacidad de desarrollar productos y servicios de calidad.

Estándares de Calidad de Software

Estándares del producto

- Se aplican al producto de software que se está desarrollando.
- Incluyen:
  - Estándares de documentos, como la estructura de los documentos de requisitos.
  - Estándares de documentación, como un encabezado de comentario estándar para una clase.
  - Estándares de codificación, que definen cómo un lenguaje de programación debería ser usado.

Estándares de Calidad de Software

Estándares del proceso

- Definen los procesos que se deben seguir durante el desarrollo de software.
- Deben encapsular buenas prácticas de desarrollo.
- Pueden incluir definiciones de especificación, diseño y validación procesos, herramientas de soporte de procesos y una descripción de los documentos que debe escribirse durante estos procesos.

Estándares de Calidad de Software

- Los estándares deben ofrecer valor, en forma de una mayor calidad del producto.
- No tiene sentido definir estándares caros en términos de tiempo y esfuerzo y solo conducen a mejoras marginales en la calidad.
- Los estándares del producto deben diseñarse de modo que puedan aplicarse y controlarse de manera rentable, y los estándares del proceso deben incluir la definición de procesos que verifican que se hayan seguido los estándares del producto.

Estándares de Calidad de Software

Para minimizar la insatisfacción y alentar el compromiso con los estándares, los gerentes de calidad que establecen los estándares deben:

- Involucrar a los ingenieros de software en la selección de estándares de productos. Si los desarrolladores entienden por qué se han seleccionado los estándares, es más probable que se comprometan a estos.
- Revisar y modificar los estándares regularmente para reflejar las tecnologías cambiantes.
- Proporcionar herramientas de software para soportar estándares.

Estándares de Calidad de Software

- Los diferentes tipos de software necesitan diferentes procesos de desarrollo, por lo que los estándares deben ser adaptables.
- Sin embargo, cuando se realizan cambios, es importante asegurarse de que estos cambios no conduzcan a una pérdida de la calidad del producto.
- El gerente del proyecto y el gerente de calidad pueden evitar los problemas de estándares inapropiados mediante una planificación de calidad cuidadosa al inicio del proyecto.

Estándar ISO 9126

Características de calidad del modelo de calidad externa e interna de ISO 9126:

- Funcionalidad.
- Confiabilidad.
- Usabilidad.
- Eficiencia.
- Mantenibilidad.
- Portabilidad.

Características de Calidad de Software

Características de calidad del modelo de calidad externa e interna de ISO 9126:

- **Funcionalidad.**
- Confiabilidad.
- Usabilidad.
- Eficiencia.
- Mantenibilidad.
- Portabilidad.

Funcionalidad

La capacidad del producto de software para **proporcionar funciones** que **cumplen con las necesidades declaradas e implícitas** cuando el software se utiliza **bajo condiciones específicas**.

Características de Calidad de Software

Características de calidad del modelo de calidad externa e interna de ISO 9126:

- Funcionalidad.
- **Confiabilidad.**
- Usabilidad.
- Eficiencia.
- Mantenibilidad.
- Portabilidad.

Confiabilidad

La capacidad del producto de software para **mantener un nivel de rendimiento específico** cuando se utiliza **bajo condiciones específicas.**

Características de Calidad de Software

Características de calidad del modelo de calidad externa e interna de ISO 9126:

- Funcionalidad.
- Confiabilidad.
- **Usabilidad.**
- Eficiencia.
- Mantenibilidad.
- Portabilidad.

Usabilidad

La capacidad del producto de software para ser **entendido, aprendido, utilizado y ser atractivo para el usuario**, cuando se utiliza **bajo condiciones específicas**.

Características de Calidad de Software

Características de calidad del modelo de calidad externa e interna de ISO 9126:

- Funcionalidad.
- Confiabilidad.
- Usabilidad.
- **Eficiencia.**
- Mantenibilidad.
- Portabilidad.

Eficiencia

La capacidad del producto de software para proporcionar un **rendimiento adecuado**, en relación con **la cantidad de recursos utilizados**, **en las condiciones establecidas**.

Características de Calidad de Software

Características de calidad del modelo de calidad externa e interna de ISO 9126:

- Funcionalidad.
- Confiabilidad.
- Usabilidad.
- Eficiencia.
- **Mantenibilidad.**
- Portabilidad.

Mantenibilidad

La capacidad del producto de software para ser **modificado**.

Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a los cambios en el entorno y en los requisitos y las especificaciones funcionales.

Características de Calidad de Software

Características de calidad del modelo de calidad externa e interna de ISO 9126:

- Funcionalidad.
- Confiabilidad.
- Usabilidad.
- Eficiencia.
- Mantenibilidad.
- **Portabilidad.**

Portabilidad

La capacidad del producto de software para **ser transferido de un entorno a otro**.

Revisiones e Inspecciones

- Las revisiones e inspecciones son actividades de control de calidad que verifican la calidad de los entregables del proyecto.
- Implican **examinar el software**, **su documentación** y **los registros del proceso** para descubrir errores u omisiones y para ver si se han seguido los estándares de calidad.
- Las revisiones e inspecciones se usan junto con las pruebas del programa como parte del proceso general de verificación y validación del software.

Revisiones

El proceso de revisión normalmente se estructura en tres fases:

1. **Actividades previas a la revisión**
2. **La reunión de revisión**
3. **Actividades posteriores a la revisión**

Revisiones

Actividades previas a la revisión

- Se trata de actividades preparatorias que son esenciales para que la revisión sea efectiva. Están relacionadas con la planificación de la revisión y la preparación de la revisión.
- La planificación de la revisión implica la creación de un equipo de revisión, la organización de la hora y el lugar para la revisión y la distribución de los documentos que se revisarán.
- Los miembros del equipo de revisión individual leen y entienden el software o los documentos y las normas pertinentes. Trabajan de forma independiente para encontrar errores, omisiones y desviaciones de los estándares.

Revisiones

La reunión de revisión

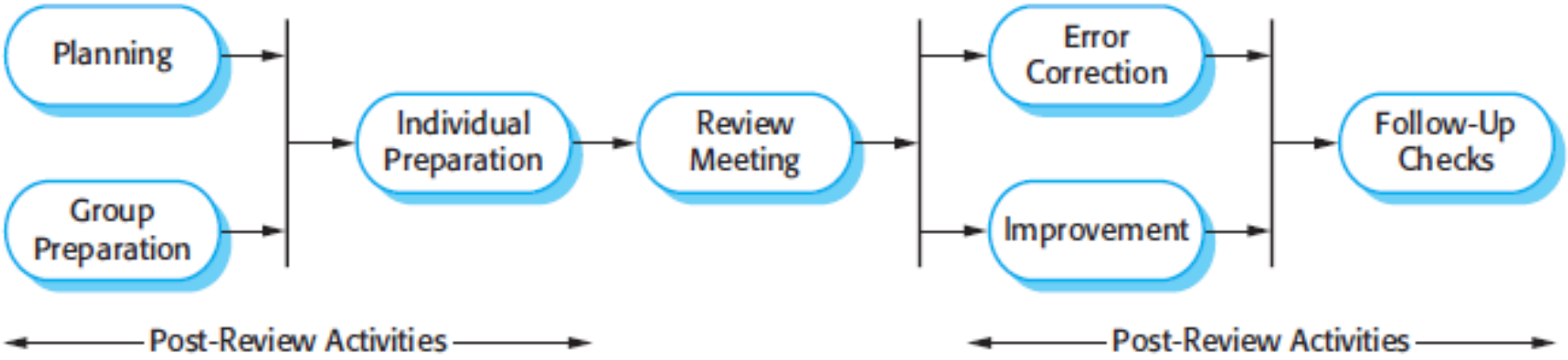
- Un autor del documento o programa que se está revisando debe "revisar" el documento con el equipo de revisión.
- Un miembro del equipo debe presidir la revisión y otro debería registrar formalmente todas las decisiones y acciones de revisión que se tomarán.
- Durante la revisión un representante es responsable de garantizar que se consideren todos los comentarios escritos.

Revisiones

Actividades posteriores a la revisión

- Después de que haya finalizado una reunión de revisión, deben abordarse los problemas planteados durante la revisión.
- Esto puede implicar la reparación de errores de software, refactorización de software para que cumpla con los estándares de calidad o la reescritura de documentos.
- Después de que se hayan realizado los cambios, el representante de la revisión puede verificar que los comentarios de la revisión se hayan tenido en cuenta.

El Proceso de Revisión de Software



Inspecciones

- Son "evaluaciones por pares" en las que los miembros del equipo colaboran para encontrar errores en el programa que se está desarrollando.
- Se pueden verificar las versiones incompletas del sistema y verificar las representaciones como los modelos UML y los casos de prueba.
- Las inspecciones del programa involucran a miembros del equipo de diferentes orígenes que hacen una cuidadosa revisión**, línea por línea**, del código fuente del programa.

Inspecciones

- Se buscan defectos y problemas que pueden ser errores lógicos, anomalías en el código o características que se han omitido del código.
- Durante una inspección, a menudo se usa una **lista de verificación** de errores de programación comunes para enfocar la búsqueda de errores.
- Esta lista de verificación puede basarse en ejemplos o en el conocimiento de defectos que son comunes en un dominio de aplicación particular.
- **Para diferentes lenguajes de programación, se utilizan diferentes listas de verificación.**

Inspecciones: Listas de Verificación

Tipo de Falla: Falla de datos

- ¿Se inicializan todas las variables del programa antes de ser utilizadas?
- ¿Han sido nombradas todas las constantes?
- ¿El límite superior de las matrices debe ser igual al tamaño de la matriz o al tamaño-1?
- ¿Hay alguna posibilidad de desbordamiento del búfer?

Inspecciones: Listas de Verificación

Tipo de Falla: Falla de control.

- Para cada enunciado condicional, ¿es correcta la condición?
- ¿Seguro que cada ciclo terminará? ¿No hay ciclos infinitos?
- ¿Están las declaraciones compuestas correctamente entre corchetes?
- En las declaraciones de casos, ¿se tienen en cuenta todos los casos posibles?
- Si se requiere un *break* después de cada caso en las declaraciones de caso, ¿se ha incluido?

Inspecciones: Listas de Verificación

Tipo de Falla: Falla de entrada/salida

- ¿Se usan todas las variables de entrada?
- ¿Todas las variables de salida tienen asignado un valor antes de su salida?
- ¿Las entradas inesperadas pueden causar daños?

Inspecciones: Listas de Verificación

Tipo de Falla: Falla de interfaz

- ¿Todas las llamadas a funciones y métodos tienen la cantidad correcta de parámetros?
- ¿Los tipos de parámetros formales y reales coinciden?
- ¿Están los parámetros en el orden correcto?

Inspecciones

Las inspecciones son muy eficaces para encontrar errores:

- Más del 60% de los errores en un programa pueden detectarse mediante inspecciones informales del programa (Fagan,1986).
- Un enfoque más formal de la inspección, basado en argumentos correctos, puede detectar más del 90% de los errores en un programa. (Mills et al., 1987)
- La tasa de detección de errores con pruebas unitarias es de aproximadamente el 25%, con inspecciones la tasa de detección de defectos es del 60% (McConnell, 2004).
- McCornnell también presenta un caso de estudio en el que la introducción de revisiones por pares provocó un aumento del 14% en la productividad y una disminución del 90% en los defectos del programa.

Inspecciones

- A pesar de su efectividad bien publicitada, muchas compañías de desarrollo de software son reacias a usar inspecciones o revisiones por pares.
- Los **procesos ágiles** rara vez usan procesos formales de inspección o revisión por pares.
- Los expertos **en programación extrema** argumentan que la programación de pares es un sustituto efectivo de la inspección, ya que es, de hecho, un proceso de inspección continuo.

Medidas y Métricas de Software

- La **medición de software** se refiere a derivar un **valor numérico o perfil** para un atributo de un componente, sistema o proceso de software.

- Al comparar estos valores entre sí y con los estándares que se aplican en toda la organización, es posible sacar conclusiones sobre la calidad del software o evaluar la efectividad de los procesos, las herramientas y los métodos del software.

Medidas y Métricas de Software

- El **objetivo a largo plazo** de la medición del software es utilizar la medición en lugar de las revisiones para emitir juicios sobre la calidad del software.
- Utilizando la medición de software, un sistema idealmente podría evaluarse utilizando un rango de métricas y podría inferirse un valor para la calidad del sistema. Si el software alcanzó un umbral de calidad requerido, entonces podría aprobarse sin revisión.
- Sin embargo, todavía se está lejos de esta situación ideal .

Medidas y Métricas de Software

Métrica de software

Una característica de un sistema de software, documentación del sistema o proceso de desarrollo que se puede medir objetivamente.

Ejemplos:

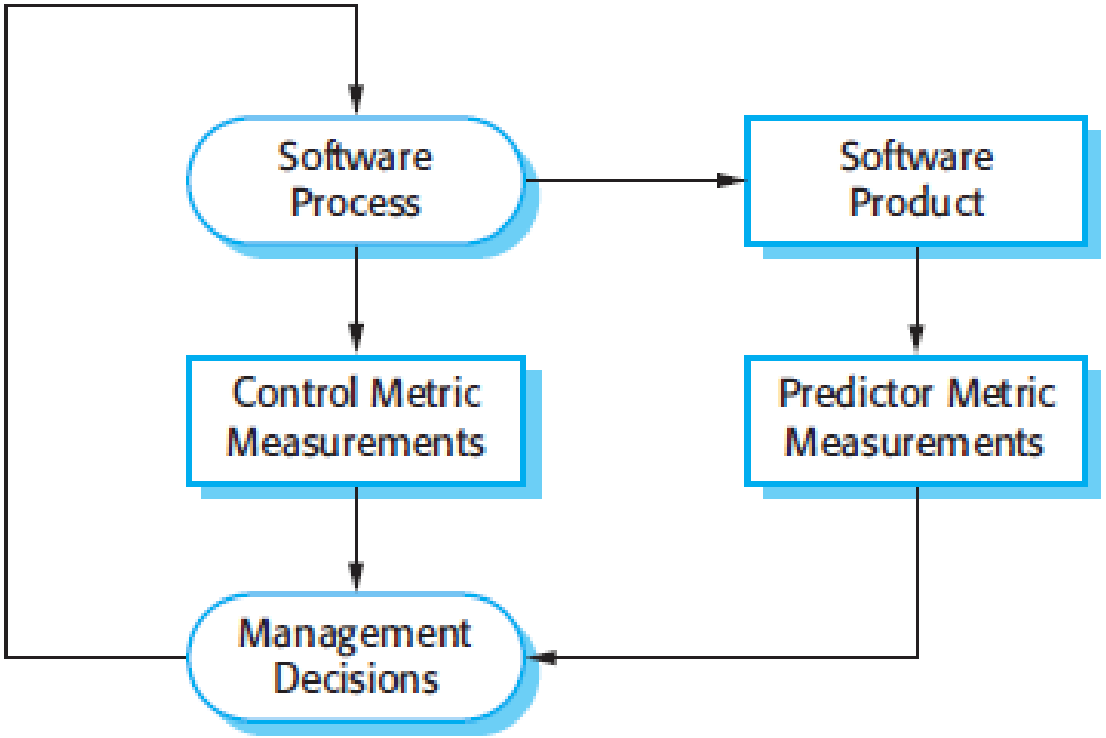
- tamaño de un producto en líneas de código;
- el índice de Fog (legibilidad de un pasaje de texto escrito);
- la cantidad de fallas informadas en un software entregado;
- la cantidad de días-persona requeridos para desarrollar un componente del sistema.

Medidas y Métricas de Software

Las métricas de software pueden ser:

- **Métricas de control**: asociadas a los procesos de software. Por ejemplo: el esfuerzo y el tiempo promedio requerido para reparar ciertos defectos informados. (Métricas del proceso).
- **Métricas de predictores**: asociadas con el software. Ayudan a predecir las características del software. Ejemplos: la longitud promedio de los identificadores en un programa, la cantidad de atributos y operaciones asociados con las clases de objetos en un diseño. (Métricas del producto).

Medidas y Métricas de Software



Medidas y Métricas de Software

Las medidas de un sistema de software se pueden usar para:

- **Asignar un valor a los atributos de calidad del sistema.**

  Al medir las características de los componentes del sistema y evaluar los atributos de calidad del sistema.

- **Identificar los componentes cuya calidad es inferior a la norma.**

  Por ejemplo, puede medir componentes para descubrir aquellos con la mayor complejidad (es más probable que contengan errores porque la complejidad los hace más difíciles de entender).

Medidas y Métricas de Software

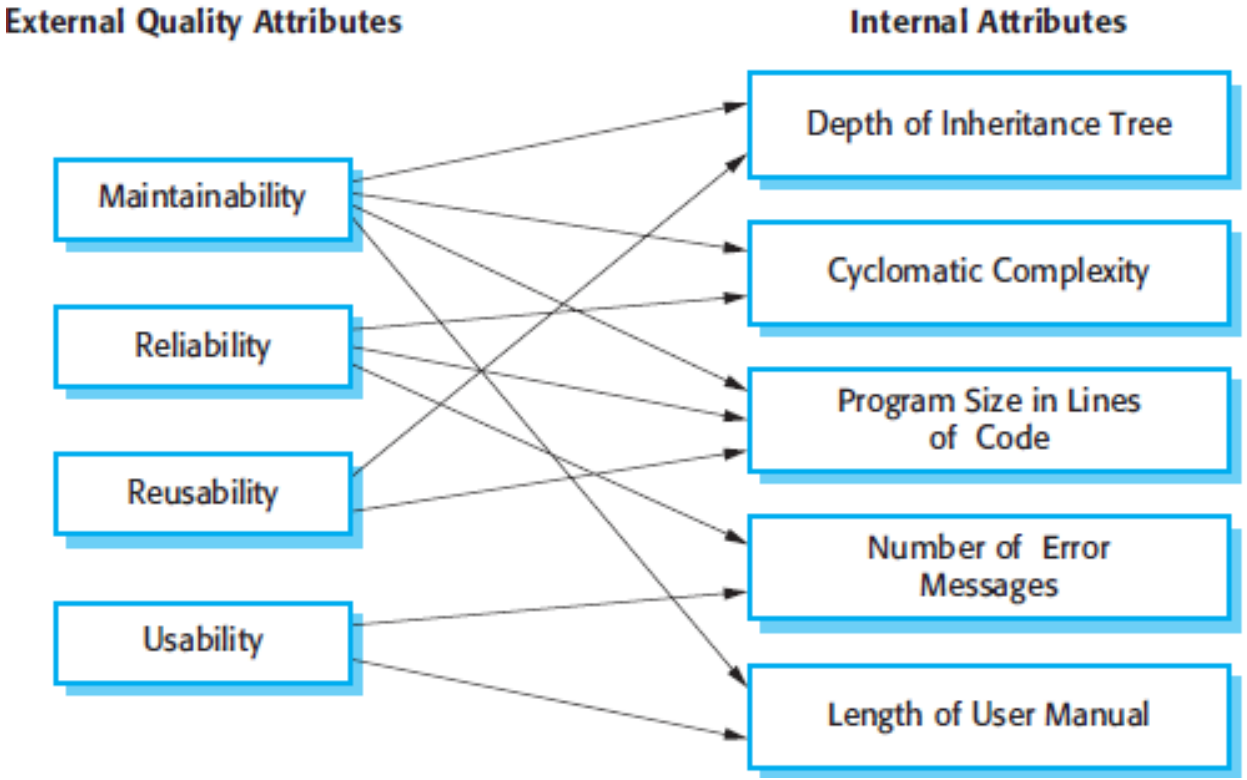
- Desafortunadamente, es difícil hacer mediciones directas de muchos de los atributos de calidad del software.
- Los atributos de calidad como la **mantenibilidad**, la **comprensibilidad** y **la facilidad de uso** son atributos externos que se relacionan con la forma en que los desarrolladores y los usuarios experimentan el software.
- Se ven afectados por factores subjetivos, como la experiencia del usuario y la educación, y por lo tanto no pueden medirse objetivamente.

Medidas y Métricas de Software

Atributos de Calidad Externos

- Los atributos de calidad como la **mantenibilidad**, la **comprensibilidad** y **la facilidad de uso** son atributos externos que se relacionan con la forma en que los desarrolladores y los usuarios experimentan el software.
- Se ven afectados por factores **subjetivos**, como la experiencia del usuario y la educación, y no pueden medirse objetivamente.
- Para juzgar estos atributos, se deben medir algunos **atributos internos** y asumir que están relacionados.

Medidas y Métricas de Software



Métricas de Producto

- Son métricas de predicción que se usan para medir los atributos internos de un sistema de software.

- Se dividen en dos clases:
  - **Métricas dinámicas**, que se recopilan mediante mediciones hechas de un programa en ejecución. Ejemplo: el tiempo que lleva completar un cálculo.
  - **Métricas estáticas**, que se recopilan mediante mediciones hechas de representaciones del sistema, como el diseño, el programa o la documentación. Ejemplos: el tamaño del código y la longitud promedio de los identificadores utilizados.

Métricas de Producto

- Generalmente, **existe una relación clara entre las métricas dinámicas y las características de calidad del software.**

- El tiempo de ejecución de funciones particulares y el requerido para iniciar un sistema se relacionan directamente con la eficiencia del sistema.
- El número de fallas del sistema y el tipo de falla se relacionan directamente con la confiabilidad del software.

Métricas de Producto

- **Las métricas estáticas, tienen una relación indirecta con los atributos de calidad.**

- Se ha intentado derivar y validar sin total éxito las relaciones entre estas métricas y la complejidad y mantenibilidad.
- Pero el tamaño del programa y la complejidad parecen ser los predictores más confiables de comprensibilidad, complejidad del sistema y mantenibilidad.

Algunas Métricas de Producto

Fan-in / Fan-out

- Fan-in es una medida del número de funciones o métodos que invocan a otra función o método (digamos X).
- Fan-out es la cantidad de funciones que la función X invoca.
- Un valor alto para fan-in significa que X está estrechamente acoplado al resto del diseño y los cambios a X tendrán un gran efecto de arrastre.
- Un valor alto para Fan-out sugiere que la complejidad general de X puede ser alta debido a la complejidad de la lógica de control necesaria para coordinar los componentes invocados.

Algunas Métricas de Producto

Longitud del código

- Es una medida del tamaño de un programa.
- Generalmente, cuanto mayor es el tamaño del código de un componente, más complejo y propenso a errores es probable que sea ese componente.
- Se ha demostrado que la longitud del código es una de las métricas más confiables para predecir la propensión a errores en los componentes.

Algunas Métricas de Producto

Longitud de identificadores

- Esta es una medida de la longitud promedio de los identificadores (nombres de variables, clases, métodos, etc.) en un programa.
- Cuanto más largos sean los identificadores, más probable será que sean significativos y, por lo tanto, más comprensible será el programa.

Algunas Métricas de Producto

Profundidad de anidamiento condicional

- Esta es una medida de la profundidad de anidamiento de sentencias if en un programa.
- Las declaraciones if profundamente anidadas son difíciles de entender y potencialmente propensas a errores.

Algunas Métricas de Producto

Índice Fog

- Es una medida de la longitud promedio de palabras y oraciones en documentos.
- Cuanto mayor sea el valor del índice Fog de un documento, más difícil será de entender el documento.

Ambigüedad de medición

- Al recopilar datos cuantitativos sobre los procesos de software, estos datos deben ser analizados para comprender su significado.
- **Es fácil malinterpretar datos y hacer inferencias que son incorrectas.**
- No se deben simplemente mirar los datos por sí solos; también se debe considerar el contexto donde se recopilan los datos.

Ambigüedad de medición

- *Un gerente decide monitorear el **número de solicitudes de cambio** enviadas por los clientes.*
  - *Se basa en la suposición de que existe una **relación entre estas solicitudes de cambio y la usabilidad** e idoneidad del producto.*
- *El gerente asume que **cuanto mayor sea el número de solicitudes de cambio, menor será la capacidad del software que satisfacer las necesidades del cliente.***

Ambigüedad de medición

- *Manejar las solicitudes de cambio y cambiar el software es costoso.*

- *Por lo tanto, la organización decide **modificar su proceso** con el objetivo de **mejorar la satisfacción** del cliente y, al mismo tiempo, **reducir los costos** de realizar cambios.*

- ***Se introducen pruebas beta** de todos los productos y las modificaciones solicitadas por los clientes se incorporan en el producto entregado.*

Ambigüedad de medición

- *Se entregan nuevas versiones de productos, desarrolladas con este proceso modificado.*

- ***En algunos casos, se reduce el número de solicitudes de cambio. En otros, se incrementa.***

- *El gerente está desconcertado y le resulta imposible evaluar los efectos de los cambios en el proceso de la calidad del producto.*

Ambigüedad de medición

- Razones por las cuales los usuarios pueden realizar solicitudes de cambio:
  - El software no es lo suficientemente bueno y no hace lo que los clientes quieren que haga. Por lo tanto, solicitan cambios para ofrecer la funcionalidad que requieren.
  - El software puede ser muy bueno y por lo tanto es ampliamente utilizado. Se pueden generar solicitudes de cambio porque hay muchos usuarios de software que piensan creativamente en cosas nuevas que podrían hacerse con el software.

Ambigüedad de medición

Aumentar la participación del cliente en el proceso puede **reducir** el número de solicitudes de cambio porque:

- Los cambios en el proceso han sido efectivos y han hecho que el software sea más útil y adecuado.
- Los cambios del proceso pueden no haber funcionado y los clientes pueden haber decidido buscar un sistema alternativo.
  El producto ha perdido participación de mercado.

Ambigüedad de medición

- Aumentar la participación del cliente en el proceso puede **aumentar** las solicitudes de cambio dado que:
  - Los cambios en el proceso pueden llevar a muchos clientes nuevos y felices que deseen participar en el proceso de desarrollo del producto.
  - Los casos de prueba no cubrieron la mayoría del uso del programa.

Ambigüedad de medición

Conclusión

- Para analizar los datos de solicitud de cambio, no solo necesita saber la cantidad de solicitudes de cambio.
- Necesita saber **quién hizo la solicitud**, **cómo se utiliza el software** y **por qué** se realizó la solicitud.
- También necesita información sobre **factores externos**, como modificaciones al procedimiento de solicitud de cambios o cambios en el mercado que puedan tener un efecto.

Material Bibliográfico

- Ian Sommerville. 2010. Software Engineering (9th ed.). Addison-Wesley Publishing Company, USA.

- Cadle, J., & Yeates, D. (Eds.). 2004. *Project management for information systems*. Pearson education.

- Epstein, D., & Maltzman, R. 2013. *Project workflow management: a business process approach*. J. Ross Publishing.